Claims

Please amend claims 1, 2, 5, 9, 11, 13, 15, 16-18, 23, 25, 27, 29, 32, 44, 46, 47, 48, and 50, cancel claims 14 and 24, and add claims 54 and 55, as follows:

1. **(Currently Amended)** A method implemented at least in part by a computing device of processing a ~~uniform~~ source language independent intermediate representation of software comprising exception handling constructs, the method comprising:

reading the ~~uniform~~ source language independent intermediate representation of software comprising exception handling constructs; wherein the ~~uniform~~ source language independent intermediate representation explicitly expresses exception handling control flow of the software; and

generating, in a computer-readable media having a tangible component, a computer-readable version of the software implementing the exception handling control flow based on the ~~uniform~~ source language independent intermediate representation.

2. **(Currently Amended)** The method of claim 1, wherein the ~~uniform~~ source language independent intermediate representation comprises:

a first instruction for expressing transfer of control to a finalization code block;

a second instruction for expressing acceptance of control transfer into the finalization code block; and

a third instruction for expressing transfer of control out of the finalization code block.

3. (Original) The method of claim 2, wherein the finalization code block comprises code related to destructor of an object.

4. (Original) The method of claim 2, wherein the finalization block comprises code related to destructor of an expression temporary object.

5. **(Currently Amended)** The method of claim 2, wherein destination operands of the second instruction [[is]] are the same as source operands of the third instruction.

6. (Original) The method of claim 2, wherein the first instruction for expressing explicit transfer of control to the finalization code block further comprises:

a label indicative of a beginning of the finalization code block to be used for expressing transfer of control to the finalization block; and

a label indicative of a continuation for control transfer after exiting the finalization code block.

7. (Original) The method of claim 2, wherein the second instruction for expressing acceptance of control transfer into the finalization code block is preceded by a label indicative of a beginning of the finalization code block and transfer of control to the finalization block is indicated by the use of the label.

8. (Original) The method of claim 2, wherein the third instruction for expressing transfer of control out of the finalization code block comprises fields for indicating different continuations for control transfer out of the finalization code block based on whether entry into the finalization code block was explicit or due to an exception.

9. **(Currently Amended)** The method of claim 1, wherein the ~~uniform~~ source language independent intermediate representation comprises:

a first instruction for catching an exception and returning an exception object related to the exception; and

a second instruction for specifying a handler for the exception based on a type value of the exception object.

10. (Original) The method of claim 9, wherein the second instruction for specifying the handler comprises:

at least one Boolean source operand for indicating the type value of the exception object;

at least one source operand indicating a label preceding a code block related to the handler to which control flow will pass if the Boolean source operand is true; and

at least one source operand indicating a label preceding a code block related to a continuation to which control flow will pass if the Boolean source operand is false.

11. **(Currently Amended)** The method of claim 1, wherein the ~~uniform~~ source language independent intermediate representation comprises:

an instruction for specifying a handler for an exception based on a type value of an exception object related to the exception, wherein a destination operand of the instruction comprises a predetermined exception object, a first source operand of the instruction comprises a label indicative of a code block related to the handler and second source operand comprises a label indicative of a code block related to a continuation.

12. (Original) The method of claim 11, wherein the instruction is operative for comparing the type value of the exception object to a type value of the predetermined exception object and if there is a match, passing control flow to the code block related to the handler label and if there is no match, then passing control flow to the code block related to the continuation label.

13. **(Currently Amended)** The method of claim 1, wherein the ~~uniform~~ source language independent intermediate representation comprises:

a first instruction for indicating entry into a try-except region; and

a second instruction for selecting one of a plurality of control flow paths for exception handling based on a type value related to the exception, wherein the plurality of control flow paths available for selection includes a path related to resumption of execution of an instruction causing the exception;

wherein the second instruction for selecting one of the plurality of control flow for exception handling comprises:

an operand indicative of the type value of the exception;

an label operand indicative of a handler code block;

an label operand indicative of a continuation code block; and

an label operand indicative of the exception causing instruction.

14. **(Canceled)**

15. **(Currently Amended)** A system for implementing ~~uniform~~ source language independent exception handling intermediate representations for multiple source code languages, the system comprising:

an intermediate language reader, implemented at least in part by a computing device of the system, for obtaining an intermediate language representation of a source code file and generating a ~~uniform~~ source language independent intermediate representation of exception handling constructs of the source code based on the intermediate language representation;

wherein the ~~uniform~~ source language independent intermediate representation explicitly expresses exception handling control flow of the source code.

16. **(Currently Amended)** The system of claim 15 further comprising a compiler for generating object code based on the ~~uniform~~ source language independent intermediate representation.

17. **(Currently Amended)** The system of claim 15, wherein the ~~uniform~~ source language independent intermediate representation of the exception handling constructs comprises a first instruction for expressing explicit transfer of control to a finalization code block, a second instruction for expressing acceptance of control transfer into the finalization code block, and a third instruction for expressing transfer of control out of the finalization code block.

18. **(Currently Amended)** The system of claim 17, wherein destination operands of the second instruction [[is]] are the same as source operands of the third instruction.

19. (Original) The system of claim 17, wherein the finalization code block comprises code related to destructor of an object.

20. (Original) The system of claim 17, wherein the finalization block comprises code related to destructor of an expression temporary object.

21. (Original) The system of claim 17, wherein the first instruction for expressing explicit transfer of control to the finalization code block further comprises a label indicative of

the beginning of the finalization code block and a label indicative of a continuation for control transfer after exiting the finalization code block.

22. (Original) The system of claim 17, wherein the second instruction for expressing acceptance of control transfer into the finalization code block is preceded by a label indicative of the beginning of the finalization code block and transfer of control to the finalization block is indicated by the use of the label.

23. (**Currently Amended**) The system of claim 17, wherein the third instruction for expressing transfer of control out of the finalization code block comprises fields for indicating different continuations for control transfer out of the finalization code block based on whether entry into the finalization code block was explicit or due to an exception, wherein the continuation for control transfer out of the finalization code block after an explicit entry matches a continuation specified by the first instruction.

24. (**Canceled**)

25. (**Currently Amended**) The system of claim 15, wherein the ~~uniform~~ source language independent intermediate representation of the exception handling constructs comprises a first instruction for catching an exception and returning an exception object related to the exception and a second instruction for specifying a handler for the exception based on a type value of the exception object.

26. (Original) The system of claim 25, wherein the second instruction for specifying the handler comprises:

at least one Boolean source operand for indicating the type value of the exception object;

at least one source operand indicating a label preceding a code block related to the handler to which control flow will pass if the Boolean source operand is true; and

at least one source operand indicating a label preceding a code block related to a continuation to which control flow will pass if the Boolean source operand is false.

27. **(Currently Amended)** The system of claim 15, wherein the ~~uniform~~ source language independent intermediate representation of the exception handling constructs comprises at least one instruction for specifying a handler for an exception based on a type value of an exception object related to the exception, wherein a destination operand of the instruction comprises a predetermined exception object, a first source operand of the instruction comprises a label indicative of a code block related to the handler and a second source operand comprises a label indicative of a code block related to a continuation.

28. (Original) The system of claim 27, wherein the instruction is operative for comparing the type value of the exception object to a type value of the predetermined exception object and if there is a match, passing control flow to the code block related to the handler label and if there is no match, then passing control flow to the code block related to the continuation label.

29. **(Currently Amended)** The system of claim 15, wherein the ~~uniform~~ source language independent intermediate representation of the exception handling constructs comprises a first instruction for indicating entry into a try-except region; and a second instruction for selecting one of a plurality of control flow paths for exception handling based on a type value related to the exception, wherein the plurality of control flow paths available for selection includes a path related to resumption of execution of an instruction causing the exception.

30. (Original) The system of claim 29, wherein the second instruction for selecting the control flow path for exception handling comprises:

an operand indicative of the type value of the exception;

an label operand indicative of a handler code block;

an label operand indicative of a continuation code block; and

an label operand indicative of the exception causing instruction.

31. (Original) The system of claim 30, wherein a handler for the first instruction for indicating entry into the try-except region is the same as a handler for the exception causing instruction.

32. **(Currently Amended)** A computer readable storage medium having a tangible component, and having stored thereon [[an]] a source language independent intermediate representation of exception handling constructs of source code, the source language independent intermediate representation of exception handling constructs comprising:

a first instruction for expressing explicit transfer of control to a finalization code block;

a second instruction for expressing acceptance of control transfer into the finalization code block; and

a third instruction for expressing transfer of control out of the finalization code block.

33. (Original) The computer readable storage medium of claim 32, wherein the first instruction for expressing explicit transfer of control to the finalization code block further comprises a label indicative of the beginning of the finalization code block and a label indicative of a continuation for control transfer after exiting the finalization code block.

34. (Original) The computer readable storage medium of claim 32, wherein the second instruction for expressing acceptance of control transfer into the finalization code block is preceded by a label indicative of the beginning of the finalization code block and transfer of control to the finalization block is indicated by the use of the label.

35. (Original) The computer readable storage medium of claim 34, wherein the transfer of control is explicit.

36. (Original) The computer readable storage medium of claim 34, wherein the transfer of control is due to an exception.

37. (Original) The computer readable storage medium of claim 32, wherein the third instruction for expressing transfer of control out of the finalization code block comprises fields for indicating different continuations for control transfer out of the finalization code block based on whether entry into the finalization code block was explicit or due to an exception.

38. (Original) The computer readable storage medium of claim 37, wherein the continuation for control transfer out of the finalization code block after an explicit entry matches a continuation specified by the first instruction.

39. (Original) The computer readable storage medium of claim 32, wherein destination operands of the second instruction are the same as source operands of the third instruction.

40. (Original) The computer readable storage medium of claim 32, wherein control flow to the finalization block is expressed by a related set of FINAL, FINALLY and ENDFINALLY instructions.

41. (Original) The computer readable storage medium of claim 32, wherein the finalization code block comprises code related to destructor of an object.

42. (Original) The computer readable storage medium of claim 32, wherein the finalization code block comprises code related to destructor of an expression temporary object.

43. (Original) The computer readable storage medium of claim 42, wherein the expression temporary object is created upon a condition being true and the control is transferred to the finalization code block upon the same condition being true.

44. **(Currently Amended)** A computer readable storage medium having a tangible component, and having stored thereon [[an]] a source language independent intermediate representation of exception handling constructs of source code, the source language independent intermediate representation of exception handling constructs comprising:

a first instruction for catching an exception and returning an exception object related to the exception; and

a second instruction for specifying a handler for the exception based on a type value of the exception object.

45. (Original) The computer readable medium of claim 44, wherein the second instruction for specifying the handler comprises:

at least one Boolean source operand for indicating the type value of the exception object;

at least one source operand indicating a label preceding a code block related to the handler to which control flow will pass if the Boolean source operand is true; and

at least one source operand indicating a label preceding a code block related to a continuation to which control flow will pass if the Boolean source operand is false.

46. **(Currently Amended)** The computer readable medium of claim 45, wherein the ~~continuation code block~~ code block related to the continuation is related to ~~another~~ a filter.

47. **(Currently Amended)** The computer readable medium of claim 45, wherein the ~~continuation related~~ code block related to the continuation comprises an unwind instruction.

48. **(Currently Amended)** A computer readable storage medium having a tangible component, and having stored thereon [[an]] a source language independent intermediate representation of exception handling constructs of source code, the source language independent intermediate representation of exception handling constructs comprising:

an instruction for specifying a handler for an exception based on a type value of an exception object related to the exception, wherein a destination operand of the instruction comprises a predetermined exception object, a first source operand of the instruction comprises a label indicative of a code block related to the handler and second source operand comprises a label indicative of a code block related to a continuation.

49. (Original) The computer readable storage medium of claim 48, wherein the instruction is operative for comparing the type value of the exception object to a type value of the predetermined exception object and if there is a match, passing control flow to the code block related to the handler label and if there is no match, then passing control flow to the code block related to the continuation label.

50. **(Currently Amended)** A computer readable storage medium having a tangible component, and having stored thereon [[an]] a source language independent intermediate representation of exception handling constructs of source code, the source language independent intermediate representation of exception handling constructs comprising:

a first instruction for indicating entry into a try-except region; and

a second instruction for selecting one of a plurality of control flow paths for exception handling based on a type value related to the exception, wherein the plurality of control flow paths available for selection includes a path related to resumption of execution of an instruction causing the exception.

51. (Original) The computer readable storage medium of claim 50, wherein the second instruction for selecting the control flow path for exception handling comprises:

an operand indicative of the type value of the exception;

an label operand indicative of a handler code block;

an label operand indicative of a continuation code block; and

an label operand indicative of the exception causing instruction.

52. (Original) The computer readable storage medium of claim 50, wherein a handler for the first instruction for indicating entry into the try-except region is the same as a handler for the exception causing instruction.

53. (Original) A system for implementing uniform exception handling intermediate representations for multiple source code languages, the system comprising:

means for reading an intermediate language representation of a source code file and generating a uniform intermediate representation of exception handling constructs of the source code based on the intermediate language representation;

wherein the uniform intermediate representation explicitly expresses exception handling control flow of the source code.

54. **(New)** The method of claim 1, wherein the source language independent intermediate representation explicitly expresses exception handling control flow of the software

by associating exception causing instructions with labels representing their related handlers, and wherein each instruction of the source language independent intermediate representation comprises a handler field.

55. (**New**) The system of claim 15, wherein the generating comprises:

reading an intermediate language code stream and associated exception handling clauses from the intermediate language representation of the source code file;

based at least in part on the exception handling clauses, determining containment relationships between protected code blocks and their associated exception handler blocks;

based on the containment relationships, assigning distinctive labels to the exception handler blocks;

based on the assignments, mapping the distinctive labels to their associated protected code blocks and exception handler blocks; and

building the source language independent intermediate representation using, at least in part, a set of source language independent exception handling instructions and the mappings.